REMARKS

Claims 1-20 are pending in the application, with claims 1, 9, and 12 being the independent claims.

Claims 1, 9, 12 and 13 have been amended.

Applicant respectfully traverses the rejection of each independent and dependent claim in the application.

Rejections under 35 U.S.C. § 103

Claims 1-20 are rejected under 35 U.S.C. 103(a) as being obvious in light of the prior art in the field.

Claims 1-20 are rejected as being unpatentable over U.S. Patent No. 6,298,434 (hereinafter "Lindwer") in view of U.S. Patent No. 6,606,743 (hereinafter "Raz"). Applicant respectfully traverses the rejection.

With respect to independent claims 1, 9 and 12, the combination of Lindwer and Raz does not teach or suggest the claimed embodiments. By Examiner's admission, Lindwer does not teach or suggest determining an entry point into shared execution code based on the stack state.

Raz does not supplement Lindwer to teach or suggest the claims. Raz does not supplement Lindwer to teach or suggest performing a stack-state-aware translation of the instruction to threaded code to determine an operand stack state for the instruction. The Action states that "The code is threaded (Raz: column 4, lines 29-31). The implementation and advantages of multithreading is well known in the art and would have been obvious to one of ordinary skill in the pertinent art at the time of the applicant's invention" (Action, pg. 6, 4 paragraph). However, this is not the definition of threaded code as it used both in the present claims and in the pertinent art. In the application, threaded code does not refer to the pseudo-parallel processing concept of multithreading, but to the computer programming concept of threaded code. Multithreading refers to the ability to run multiple executables on a single processor through the use of threads, wherein each executable is contained within a thread and the processor is able to switch its execution rapidly between threads to create the appearance of parallel processing. This is the definition of thread used in Raz. Raz states "The smart DMA can also optionally be used to accelerate thread context switching by

moving data into and out of the cache, as required." (Raz: column 4, lines 29-31). In the present

claims, threaded code does not refer to multithreading. Rather, threaded code is a term of art in

computer programming for a technique for producing very compact code, which may be composed

entirely of subroutine calls written as memory addresses containing the memory locations of the

subroutine being called. It is an entirely distinct concept from multithreading. Raz discloses the use

of multithreading, but does not teach or suggest the use of threaded code.

Claims 1, 9 and 12 are presently amended. The Action cites the use of the term "shared

execution code" as allowing for a broad interpretation of the claims. The term "shared" has been

replaced with the term "cascading," and further description of "cascading execution code" has been

added to the claims. "Cascading" is used to reflect the multi-tiered nature of the execution code in

the present claims, wherein execution may begin at any discrete tier of the code based on the current

stack-state. For example, for an iadd instruction:

[1] Stack state 1 for iadd: Stack-operations-A.1

[2] Stack state 2 for iadd: Stack-operations-A.2

[3] Stack state 3 for iadd: Stack-operations-A.3

...

[n-1] Stack state n-1 for iadd: Stack-operations-A.n-1

[n] Stack state n for iadd: Stack-operations-A.n

A java bytecode is sent through the stack-state aware translator and changed into threaded code.

The operand stack state of each instruction is used to create the threaded code, such that when an

bytecode instruction, for example, an iadd instruction, is called from the currently executed code,

the stack state aware translation of the instruction will include the operand stack state, which will

allow the iadd instruction in the cascading execution code to be entered at the proper tier given the

operand stack state of the instruction. If an iadd operation is called when the operand stack is in

stack state 1, for example, the iadd instruction will be entered at the tier [1]. If the operand stack is

in stack state 3, the iadd instruction will be entered at tier 3. Execution in the cascading execution

code will then proceed down the tiers.

It should be noted that this amendment does indicate that the "cascading execution code" is not shared. The "cascading execution code" may still a shared code, as it may be reusable by and commonly shared among any number of executable programs. Additionally, the "cascading execution code" may be shared among multiple processors on the same computer or across different computers, or by any other method of sharing in accordance with the broad definition of "shared" cited in the Action.

Applicant respectfully requests that the rejection of independent claims 1, 9, and 12 be withdrawn on at least the basis of this amendment, as neither Raz nor Lindwer disclose a cascading execution code, and that the finality of the Action be withdrawn and prosecution continued on the claims as amended.

Lindwer does not teach all of the limitations of the present claims and the combination of Lindwer with Raz does not overcome the deficiencies of Lindwer. Independent claims 1, 9, and 12 are allowable over Lindwer in view Raz.
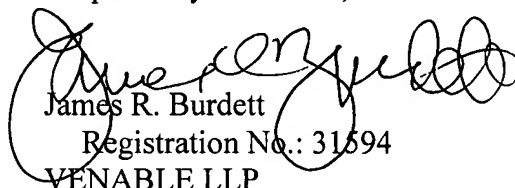
Claims 2-8, 10, 11, 13-20 depend from the independent claims which are allowable over Lindwer and Raz as discussed above.

Conclusion

All of the stated grounds of rejection have been properly traversed. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding rejections and that they be withdrawn. Applicants believe that a full and complete reply has been made to the outstanding Office Action and, as such, the present application is in condition for allowance. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is hereby invited to telephone the undersigned at the number provided.

Dated: April 9, 2007

Respectfully submitted,

James R. Burdett
    Registration No.: 31,594
VENABLE LLP
P.O. Box 34385
Washington, DC 20043-9998
(202) 344-4000
(202) 344-8300 (Fax)
Attorney/Agent For Applicant